On Using "Stochastic Learning on the Line" to Design Novel Distance Estimation Methods for *Three-Dimensional Environments*

Jessica Havelock[1], B. John Oommen[1,2(✉)], and Ole-Christoffer Granmo[2]

[1] School of Computer Science, Carleton University, Ottawa, Canada
oommen@scs.carleton.ca
[2] Centre for Artificial Intelligence Research, University of Agder, Grimstad, Norway

Abstract. We consider the unsolved problem of Distance Estimation (DE) when the inputs are the x and y coordinates (i.e., the latitudinal and longitudinal positions) of the points under consideration, *and* the elevation/altitudes of the points specified, for example, in terms of their z coordinates (3DDE). The aim of the problem is to yield an accurate value for the real (road) distance between the points specified by *all the three* coordinates of the cities in question (This is a typical problem encountered in a GISs and GPSs.). In our setting, the distance between *any pair of cities* is assumed to be *computed* by merely having access to the coordinates and *known* inter-city distances of a *small subset* of the cities, where these are also specified in terms of their 3D coordinates. The 2D variant of the problem has, typically, been tackled by utilizing parametric functions called "Distance Estimation Functions" (DEFs). To solve the 3D problem, we resort to the Adaptive Tertiary Search (ATS) strategy, proposed by Oommen *et al.*, to affect the learning. By utilizing the information provided in the 3D coordinates of the nodes and the true road distances from this *subset*, we propose a scheme to estimate the inter-nodal distances. In this regard, we use the ATS strategy to calculate the best parameters for the DEF. While "Goodness-of-Fit" (GoF) functions can be used to show that the results are competitive, we show that *they are rather not necessary to compute the parameters*. Our results demonstrate the power of the scheme, even though we completely move away from the traditional GoF-based paradigm that has been used for four decades. Our results conclude that the 3DDE yields results that are far superior to those obtained by the corresponding 2DDE.

Keywords: Road distance estimation · Estimating real-life distances · Learning Automata · Adaptive Tertiary Search · Stochastic Point Location

F. Wotawa et al. (Eds.): IEA/AIE 2019, LNAI 11606, pp. 39–49, 2019.
https://doi.org/10.1007/978-3-030-22999-3_4

1 Introduction

In this paper, we consider the problem of estimating the real (road) distances between points (cities), where the inputs are the x and y coordinates of the points under consideration, and their z coordinates. We refer to this problem as 3DDE. This problem is much more complicated than the corresponding 2D problem that has been studied for a few decades (the 2DDE), because in hilly terrains, estimating the road distances is more complex than on flat domains.

Historically, the input to Distance Estimation (DE) problems are, typically, the start and end locations in the form of x and y co-ordinates of the locations in the Cartesian plain, or the latitude and longitude in the geographic region. However, determining the actual "road distances" (the physical distance to be traveled on the "roads" built in the community) for an area, is more challenging. These road distances, also synonymously known as traveling distances or "true" distances, can depend on the network, the terrain, the geographical impediments like rivers or canyons, and of course, the direct distance between the respective points – which serves as a lower bound for the "true" distances. The 2DDE and 3DDE problems involve finding the best estimator for these true distances.

This problem has been studied for over four decades, and its solutions have been used in many practical applications, such as in developing vehicle scheduling software, vehicle routing, and in partitioning districts for fire-fighters [1,2,7].

Legacy Methods: Distance Estimation Functions. To historically do DE, one typically resorts to Distance Estimating *Functions* (DEFs) [5] which are simultaneously good estimators, characterized by low computations. All these DEFs involved parameters whose values are obtained by a "training" phase to best fit the data of the system being characterized, with some "true" known *a priori* road distances. The accuracy of the estimations depends on the DEF, the system and the available data.

Our Proposed Approach. In this paper, we apply a new method for determining the DEF, earlier pioneered in [3], namely the Adaptive Tertiary Search (ATS) [6]. The ATS uses Learning Automata (LA) to perform a stochastic search "on a line" to determine the parameter sought for. Our most "daring" step is to *completely move away from invoking Goodness-of-Fit (GoF) criteria for the DEFs* as has been done for more than four decades [8,9].

2 Distance Estimation: Core Concepts

DE is a topic that has been extensively studied for more than four decades. Due to space limitations, it is impossible to survey the field here. But we refer the reader to [3] and [4], where it has been surveyed in greater detail.

DE is, typically, done by determining the appropriate DEF, which is a mapping from $R^d \times R^d$ to R. It returns the estimate of the true distance. The inputs to the DEF are the locations of the two points, and it produces an estimate of the distance between them by incorporating the set of parameters into the DEF. This set of parameters is *learnt* so that the DEF best represents the space.

Definition 1. *A **Distance Estimation Function (DEF)** is defined as a function* $\pi(P_1, P_2|\Lambda) : R^d \times R^d \longrightarrow R$*, in which* $P_1 = \langle x_1, x_2, ..., x_d \rangle$ *and* $P_2 = \langle y_1, y_2, ..., y_d \rangle$ *are points in* R^d*, and* Λ *is a set of parameters characterizing* π*, learnt using a set of training points with known true inter-point distances.*

This set, Λ is, typically, learnt by minimizing a GoF function which is used to measure how well a region is represented by the DEF. Central to DE is the concept of GoF functions which are measures of how good a DEF estimates the true (but unknown) distances. Several GoF functions have been utilized in the literature, and the most common one is the sum of Square Deviation (SD).

The most common types of DEFs are those based on the family of L^p norms, traditionally used for computing distances:

$$L_p(X) = \left(\sum_{i=1}^{n} (|x_i|^p \right)^{1/p}. \tag{1}$$

The various well-known L^p norms have been used as stepping stones to design DEFs as seen in [5]. The input to these functions are the co-ordinates of the input vectors, X and Y. These DEFs are first trained on the subset of the co-ordinates of the cities and *their* known (region-based) true distances, and this yields the "best" parameters for the DEF given the training data. Thereafter, the DEF can be used for DE for other unknown inter-city distances.

3 The Adaptive Tertiary Search and Its Use in DE

The solution that we propose for DE is based on a scheme relevant to the Stochastic Point Location (SPL) problem. To formulate the SPL, we assume that there is a Learning Mechanism (LM) whose task is to determine the optimal value of some variable (or parameter), λ. We assume that there is an optimal choice for λ - an unknown value, say $\lambda^* \in [0, 1]$. As in [3], in this paper, we shall use the ATS [6] to solve the DE problem, although we could have used other solutions.

Table 1. The decision table for the ATS scheme.

O^1	O^2	O^3	New sub-interval
Inside	Left	Left	Δ^1
Left	Left	Left	Δ^1
Right	Inside	Left	Δ^2
Right	Left	Left	$\Delta^1 \cup \Delta^2$
Right	Right	Inside	Δ^3
Right	Right	Left	$\Delta^2 \cup \Delta^3$
Right	Right	Right	Δ^3

To determine λ^* within the resolution of accuracy, the original search interval is divided into three equal and disjoint subintervals, Δ^i, where $i = 1...3$. The subintervals are searched using a two-action LA. The LA returns the $\lambda(n)$, the estimated position of λ^* from that subinterval $O^i \in \{Left, Right, Inside\}$. From these, a new search interval is obtained based on the table given in Table 1. This is repeated until the search interval is smaller than the resolution of accuracy. The search interval will yield the required resolution within a finite number of epochs because the size of the search interval is decreasing [6]. After the interval is small enough, the midpoint of the final interval is the estimate for λ^*.

3.1 Updating Search Intervals

Let us first consider the case where the DEF has two parameters, say k and p. The strategy for our search will be to use the ATS to determine the best value for k and p, say k^* and p^*, respectively. It is crucial that the *order* of updating the search intervals in the k and p spaces is considered when determining these multiple parameters. If this is not done correctly, it may result in the premature reduction of a search interval. The order of executing the searching, and the pruning of the intervals must also be maintained while searching for the two parameters, k and p, simultaneously. Thus, all the subintervals must be searched before the intervals are updated simultaneously, as shown in [3] and [4].

The set of LA operate in the same manner as in [6], except for how it deals with the additional parameters. When the LA is learning information about how it should update the value for k, it uses values of p from within its *current* search interval and vice versa. As a result, each LA operates with the knowledge of the *current* search interval of *all* the other parameters, as explained in [3] and [4].

This process of searching for multiple parameters can be done in parallel by assuming that for each learning loop, the other parameter's value is either the maximum or the minimum of its current search interval. This is a consequence of the monotonicity of the DEFs, as discussed in Sect. 3.3.

3.2 The Corresponding LA

Each LA is provided with two inputs, namely the parameter that it is searching for, and all the search intervals, and yields as its output the relative location of the parameter in question. It does this by producing a decision (Left, Right or Inside) based on *its* final belief after communicating with *its* Environment.

The LA starts out with a uniform belief, 50%, for both "Left" and "Right". It then makes a decision based on its current belief. If the decision is "Left", then the LA picks a point in the left half of the interval at random; otherwise (i.e., the decision is "Right") the point is chosen from the right half of the interval. Once the decision is made, the LA asks the Environment for a response. The LA uses the Linear Reward-Inaction (L_{RI}) update scheme, and so the current belief is only updated if the Environment's response is positive.

The LA and the Environment repeat this loop for a large number, say N_∞, of iterations. After they are done communicating, the LA produces its output as

per the LA algorithm briefly described below, but omitted here in the interest of space. It is found in [3] and [4]. If the LA's belief of "Right" is greater than $1-\epsilon$, the parameter in question is to the right side of the current search interval, and so its output is "Right". Conversely, if the belief of "Left" is greater than $1-\epsilon$, the LA's final decision is "Left". If neither of these cases emerge, the LA does not have a belief greater than $1-\epsilon$ that the parameter is to the "Right" or "Left", and in this case, the LA decides that the parameter's optimal value is "Inside" the present interval. Again, the entire algorithm is formally given in [3] and [4] (omitted here in the interest of space).

3.3 The Corresponding Environment

Each LA requires feedback from a specific Environment. This feedback informs the LA if it has made the correct decision, i.e., choosing the right or left half of the subinterval. It is easy to obtain this answer because it only involves a single parameter at a time. To further explain this, consider the DEFs below:

$$F(k,p) = k \cdot F_1(X_1, X_2, p), \text{and where,} \tag{2}$$

$$F_1(X_1, X_2, p) = \left(\sum_{i=1}^{d} |x_{1i} - x_{2i}|^p \right)^{1/p}. \tag{3}$$

Although nothing specific can be said about the monotonicity characteristics of $F(k,p)$, we see from Eq. (2) that by virtue of the fact that it is always positive and that it can be factored, it is monotonically *increasing* with k for any fixed value, p. Similarly, from Eq. (3), since $\mathrm{F}_1(X_1, X_2, p)$ is not a function of k, it is monotonically *decreasing* with p for any fixed value of k. These properties allow the Oracle to respond accordingly when finding k, and for the corresponding LA to move in the desired direction (i.e., "Left" or "Right") in the space that only involves the single parameter k. The contrary monotonicity properties allow the Oracle to respond according to a corresponding algorithm (*EnvironmentResponseP*, explained in[1] in [3] and [4]) when determining p, and for the corresponding LA to move in the desired direction in the space that involves only p.

4 Testing and Results: 3-Dimensional Environments

As opposed to the theory and results presented in the predecessor paper [3], in this paper, we have considered the problem of DE in three dimensions, i.e., 3DDE. To permit us to do testing of 3DDE, since real-life data is currently not available[2], we have resorted to utilizing "realistic" artificially-constructed data. Data of this type will give us a better understanding of how the DE method

[1] The proofs of the relevant claims are also included in these publications.

[2] The use of a third dimension (the altitude) could be especially beneficial if the region in question has a predominantly hilly or mountainous terrain, as in Perugia, Italy, or Zermatt and Switzerland.

behaves. Using the artificially-constructed data, we have compared the results from the 3DDE using the two dimensional method presented in [3]. From these comparisons, we are able to show which method is superior in these situations.

4.1 Experimental Setup

To compare the 3DDE with the typical two dimensional DE (2DDE), we used the ATS in combination with the weighted L^p DEF chosen because it has been well studied for DE and is very versatile. It can also out-perform many other DEFs in networks of differing structures. Finally, it possesses a simplicity that generally performs well when one considers that it is a DEF with only two variables. The metrics we used were the Normalized Absolute-value Difference (NAD), the sum of Square Deviation (SD), the Relative Absolute-value Difference (RAD) and Expected Percent (EP) errors which are recommended in the literature.

All of the data sets consisted of 100 points, representing the cities. This value was chosen based on the common data set sizes in the literature. The distance between the intermediate points, D, was another parameter that had to be defined. We used the value $D = 0.05$, which was based on the largest rate of change of the terrain and the desired accuracy for the inter-point distances. This inter-point distance was small enough to account for the features on all the surfaces considered in this paper. Further, both the 2D and 3DDE methods used the same data sets. The only difference for the 2DDE method was that it did not incorporate the third dimension.

For both the noiseless and the noisy data, there were three types of surfaces (given below) used to construct the data sets. The first type was a single hill, an example of which is shown in Fig. 1 (left). Five different single Gaussian hill surfaces, $N(0.5, \sigma)$, were constructed to determine the accuracy of the DE of both "easy" and "hard" terrains. The parameters for these surfaces are shown in Table 2. The second type of surface used was a valley. This consisted of two Gaussian hills, $N(0, \sigma_1) + N(1, \sigma_2)$ located at opposite corners of the terrain, which had a 1×1 unit base. An example of this is shown in Fig. 1 (center). Again, the steepness of the valley was varied according to the values in Table 2. The last surface consisted of two Gaussian hills located side-by-side, $N(\mu_1, \sigma_1) + N(\mu_2, \sigma_2)$. This is, clearly, the most complete surface and is shown in Fig. 1

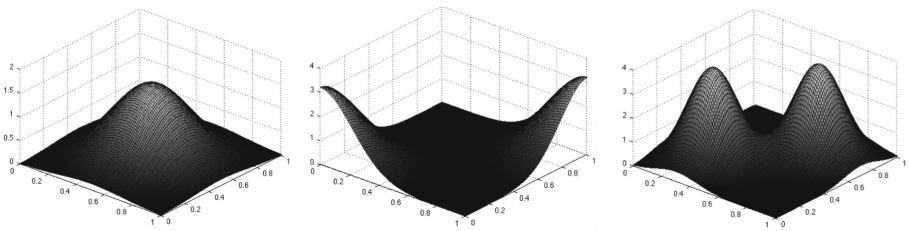

Fig. 1. Example of 3D Surfaces used in our experiments.

(right). The differences between the two hill surfaces are outlined in Table 2. For this scenario, the steepness was also changed.

4.2 Results for Noisy Surfaces

The results of the 3DDE was compared to the 2DDE over 20 runs. Further, the level of noise used was different for each run.

Table 2. Parameters of the 3D testing surfaces used in our experiments.

Surface:	Single hill		Valley		Two hills	
Number	Mean (μ)	Variance (σ^2)	Means (μ_1, μ_2)	Variances (σ_1^2, σ_2^2)	Means (μ_1, μ_2)	Variances (σ_1^2, σ_2^2)
1	(0.5)	(0.3)	(0.0, 1.0)	(0.02, 0.02)	(0.2, 0.8)	(0.09, 0.09)
2	(0.5)	(0.2)	(0.0, 1.0)	(0.05, 0.05)	(0.2, 0.8)	(0.09, 0.07)
3	(0.5)	(0.1)	(0.0, 1.0)	(0.09, 0.09)	(0.2, 0.8)	(0.07, 0.07)
4	(0.5)	(0.09)	(0.0, 1.0)	(0.12, 0.12)	(0.2, 0.8)	(0.07, 0.05)
5	(0.5)	(0.08)	(0.0, 1.0)	(0.15, 0.15)	(0.2, 0.8)	(0.05, 0.05)

Single Hill Surfaces: Table 3 shows the errors for the surfaces containing a single hill. These surfaces are in order from the flattest to the steepest. The first single hill surface had a value of $\sigma^2 = 0.3$. This was a relatively flat hill where the z-component was irrelevant, and as a result, the 2DDE marginally outperformed the 3DDE. The 3DDE method had a slight improvement to about 8.8% when $\sigma^2 = 0.2$, while the 2DDE error increased to over 9% error. When σ^2 was increased to values greater than or equal to 0.1, the 2DDE produced an

Table 3. Results for 20 runs of 2D ATS and 3D ATS on noisy single hill surfaces.

Name σ^2	Hill-1 0.3	Hill-2 0.2	Hill-3 0.1	Hill-4 0.09	Hill-5 0.08
	3D	3D	3D	3D	3D
SD	3.05	3.13	8.01	12.26	23.16
NAD	40.09	38.46	56.36	63.70	77.81
RAD	0.0950	0.0871	0.1369	0.1726	0.2135
EP	0.0922	0.0884	0.1296	0.1464	0.1789
	2D	2D	2D	2D	2D
SD	2.24	3.78	32.89	30.91	79.61
NAD	36.05	44.28	145.42	147.94	209.79
RAD	0.0787	0.0933	0.2722	0.2608	0.3627
EP	0.0829	0.1018	0.3343	0.3401	0.4823

error greater than 26%. For the 3DDE, all the errors were less than 21%. One observes that errors obtained increased with the steepness of the hill.

Valley Surfaces: The results of the 2DDE and 3DDE on the valley surfaces with noisy distances are summarized in Table 4. For the valley surfaces, the 2DDE and the 3DDE have the closest performance on the valley with the steepest sidewalls. The 2DDE had errors of 13%, and the 3DDE had an expected error for an estimated distance and the overall error of 11% and 15% respectively. As the valleys flattened out, the RAD and EP errors for the 2DDE increased to 35% and 40% respectively. Thereafter, the 2DDE improved as the valleys continued to flatten. For the flattest valley, the 2DDE had errors of 18% and 20%. The 3DDE decreased to a maximum of 5%. When $\sigma^2 > 0.02$, the errors were all between 12% and 9%. The reader must observe the superiority of the 3DDE.

Table 4. Results for 20 runs of 2D ATS and 3D ATS on various noisy valley surfaces.

Name σ^2	Valley-1 0.02	Valley-2 0.05	Valley-3 0.09	Valley-4 0.12	Valley-5 0.15
	3D	3D	3D	3D	3D
SD	21.63	5.93	7.39	4.29	3.56
NAD	49.40	46.24	48.27	45.99	42.40
RAD	0.1469	0.1128	0.1196	0.1012	0.0915
EP	0.1136	0.1063	0.1110	0.1057	0.0975
	2D	2D	2D	2D	2D
SD	17.23	31.42	53.60	26.22	12.80
NAD	57.95	146.90	174.09	136.31	86.96
RAD	0.1318	0.3036	0.3593	0.2863	0.1784
EP	0.1332	0.3377	0.4002	0.3133	0.1999

Table 5. Results for 20 runs of 2D ATS and 3D ATS on noisy surfaces with two hills.

Name (σ_1^2, σ_2^2)	Two Hills-1 $(0.09, 0.09)$	Two Hills-2 $(0.09, 0.07)$	Two Hills-3 $(0.07, 0.07)$	Two Hills-4 $(0.07, 0.05)$	Two Hills-5 $(0.05, 0.05)$
	3D	3D	3D	3D	3D
RAD	0.1728	0.1635	0.1637	0.2238	0.2546
EP	0.1553	0.1552	0.1529	0.1957	0.2239
	2D	2D	2D	2D	2D
RAD	0.3236	0.3810	0.6673	0.7219	0.8441
EP	0.4029	0.4738	0.7499	0.8736	1.0960

Two Hill Surfaces: Table 5 shows the results for the noisy distances on the surface with two hills. The surfaces are arranged, based on the mean height of the two hills from the shortest to the highest.

The 3DDE had RAD and EP errors ranging from 25% to 15%. The terrains with the smallest error contained hills where $\sigma_1^2 = 0.09$, $\sigma_2^2 = 0.07$ and $\sigma_1^2 = 0.07$, $\sigma_2^2 = 0.07$. The largest error of 25% for the 3DDE was caused by estimating distances for the surface with the two steepest hills, $\sigma_1^2 = 0.05$ and $\sigma_2^2 = 0.05$. This surface also forced the 2DDE to produce its largest error of 109%, which is actually unacceptable, and which clearly advocates the use of the 3DDE domain. In fact, the smallest error that the 2DDE produced for the surfaces with two hills was 32%, which was for the flattest hills.

4.3 Discussion

Results for Noisy Surfaces: The discussion, for results for noisy surfaces, will be focused on the RAD errors. These errors, for both the 2DDE and the 3DDE, are plotted against the steepness of the surface in Fig. 2(a)–(c). The flatter surfaces are on the right of each plot while the steeper surfaces are on the left.

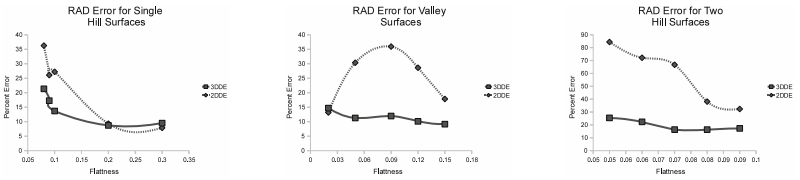

Fig. 2. Plot of RAD errors versus the steepness of the terrain for (a) the noisy single hill data sets, (b) the noisy valley data sets, and (c) the noisy two hill data sets.

Single Hill Surfaces: Both the 3DDE and the 2DDE had the same trend for the single hill surfaces. They both perform better for the flatter surfaces.

Valley Surfaces: The 3DDE had little variance in the errors. They ranged from 15% to 9%, with the steepest valley having the largest errors, and the flattest valley having the lowest errors. All of the noisy 3DDE errors were larger than their noiseless counterparts. The 2DDE did not, however, follow a linear trend. It performed best on the steepest valley, and the flattest valley.

Two Hill Surfaces: Just as for the noiseless data, the surfaces with two hills were the hardest for both methods to produce accurate estimates. Both performed best on the flatter hills, and worst on the steepest.

To get a closer look at where the errors were coming from, Fig. 4 shows the distribution of errors on noiseless and noisy data sets respectively. Each figure contains 500 points. The errors are represented by how red each point is. The black points contain the smallest errors while the red, the highest (Fig. 3).

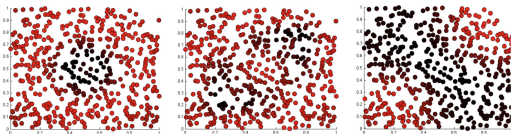

Fig. 3. The distribution of the errors for the noiseless 3D data sets. In 3a (left) the error distribution for a noiseless hill surface is shown, 3b (center) shows the error distribution for two hills, and 3c (right) shows the distribution for a valley. (Color figure online)

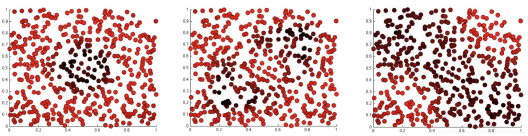

Fig. 4. The distribution of the errors for the noisy 3D data sets. In 4a (left) the error distribution for a noisy hill surface is shown, 4b (center) shows the error distribution for two hills, and 4c (right) shows the distribution for a valley. (Color figure online)

The trends for both the noiseless and noisy data sets were the same. For surfaces with one and two hills, the ATS favored the tops of the hills. As opposed to this, for the valley surfaces, the ATS produces the best estimates at the bottom of the valley.

5 Conclusions

In this paper, we have considered the Distance Estimation (DE) problem that has been studied for almost four decades. This paper has generalized the problem studied to-date to also permit the input to be the elevation/altitudes of the points – their z coordinates (3DDE). To the best of our knowledge, this is the first and pioneering attempt to solving 3DDE, the DE problem in which the inputs are all the three coordinates of the cities. Further, we have assumed that the distance between *any pair of cities* is to be *computed* by merely having access to the coordinates and *known* inter-city distances of a *small subset* of the cities.

Our solution departs from the legacy methods in that we ignore the use of so-called "Goodness-of-Fit" (GoF) functions. Rather, we have used the field of Learning Automata (LA) and in particular, the Adaptive Teriary Search (ATS) used to solve the Stochastic Point Location (SPL) problem. The results obtained also lead us to conclude that the pioneering application of the ATS to the 3DDE yields results that are far superior to those obtained by the corresponding 2DDE.

References

1. Brimberg, J., Love, R.F., Walker, J.H.: The effect of axis rotation on estimation. Eur. J. Oper. Res. **80**, 357–364 (1995)
2. Erkut, H., Polat, S.: A simulation model for an urban fire fighting system. OMEGA - Int. J. Manag. Sci. **20**(4), 535–542 (1992)
3. Havelock, J., Oommen, B.J., Granmo, O.-C.: Novel distance estimation methods using "stochastic learning on the line" strategies. IEEE Access **6**, 48438–48454 (2018). https://doi.org/10.1109/ACCESS.2018.2868233
4. Havelock, J., Oommen, B.J., Granmo, O.-C.: Novel distance estimation methods for 3D spaces using "stochastic learning on the line" strategies. Unabridged version of this paper (2019)
5. Love, R.F., Morris, J.G.: Modelling inter-city road distances by mathematical functions. Oper. Res. Q. **23**(1), 61–71 (1972)
6. Oommen, B.J., Raghunath, G.: Automata learning and intelligent tertiary searching for stochastic point location. IEEE Trans. Syst. Man Cybern. **28**(6), 947–954 (1998)
7. Oommen, J., Altınel, I.K., Aras, N.: Discrete vector quantization for arbitrary distance function estimation. IEEE Trans. Syst. Man Cybern. **28**(4), 496–510 (1998)
8. Ortega, F.A., Mesa, J.A.: A methodology for modelling travel distances by bias estimation. Sociedad de Estadistica e lnvestigacion Operativa **6**(2), 287–311 (1998)
9. Uster, H., Love, R.F.: Application of a weighted sum of order p to distance estimation. IIE Trans. **33**(8), 675–684 (2001)